Ahmet Emre Tümer - 152120171021

Introduction to Artificial Intelligence Lab

Lab 2: Queue, stack.

1. Implement the following data structures:
   a. Queue with all its methods (enqueue, dequeue, sort, etc.)
   b. Stack with all its methods (push, pop, sort, etc.)
   c. Both structures must have `delete_duplicates()`, `list()`, `reverse_list()` functionalities with minimum time complexity and memory management.
   You can extend the model count likewise.

   Question weights are as follows: a=3, b=3 and c=4.

Answers:

a. Queue with all its methods (enqueue, dequeue, sort, etc.)

```
#include <stdio.h>
#include <iostream>
#include <climits>
#include <queue>
using namespace std;

// queue sınıfı
class Queue {
public:
	int front, rear, size;
	unsigned capacity;
	int* array;
};

// Sıra oluşturmak için kullanılan fonksiyon kapasite = 0
Queue* createQueue(unsigned capacity)
{
	Queue* queue = new Queue();
	queue->capacity = capacity;
	queue->front = queue->size = 0;

	queue->rear = capacity - 1;
	queue->array = new int[queue->capacity];
	return queue;
}

// Sıranın dolu olup olmadığını sorgulayan fonksiyon
int isFull(Queue* queue)
{
	return (queue->size == queue->capacity);
}

// Sıranın boş olup olmadığını sorgulayan fonksiyon
int isEmpty(Queue* queue)
{
	return (queue->size == 0);
}
```

```
// Sıraya ekleme yapma fonksiyonu
void enqueue(Queue* queue, int item)
{
	if (isFull(queue))
		return;
	queue->rear = (queue->rear + 1)
		% queue->capacity;
	queue->array[queue->rear] = item;
	queue->size = queue->size + 1;
	cout << item << " enqueued to queue" << endl;
}

// Sıradan eleman silme fonksiyonu
int dequeue(Queue* queue)
{
	if (isEmpty(queue))
		return INT_MIN;
	int item = queue->array[queue->front];
	queue->front = (queue->front + 1)
		% queue->capacity;
	queue->size = queue->size - 1;
	return item;
}

// Sıranın önündeki eleman
int front(Queue* queue)
{
	if (isEmpty(queue))
		return INT_MIN;
	return queue->array[queue->front];
}

// Sıranın sonundaki eleman
int rear(Queue* queue)
{
	if (isEmpty(queue))
		return INT_MIN;
	return queue->array[queue->rear];
}

//Sırayı listeleme
void list_queue(Queue* queue)
{
	while (true)
	{
		if (isEmpty(queue))
		{
			break;
		}
		else
		{
			cout << front(queue) << " ";
			dequeue(queue);
		}
	}
	cout << endl;
}

// Sıralama fonksiyonu
void sortQueue(Queue* queue)
{
```

```
	if (isEmpty(queue))
		return;

	int temp = front(queue);

	dequeue(queue);

	sortQueue(queue);

	pushInQueue(queue, temp, queue->size);
}

void pushInQueue(Queue* queue,int temp, int qsize)
{

	if (isEmpty(queue) || qsize == 0) {
		enqueue(queue, temp);
		return;
	}

	else if (temp <= front(queue)) {
		enqueue(queue, temp);

		FrontToLast(queue, qsize);
	}
	else {
		enqueue(queue, front(queue));
		dequeue(queue);
		pushInQueue(queue, temp, qsize - 1);
	}
}

int main()
{
	Queue* queue = createQueue(1000);

	enqueue(queue, 10);
	enqueue(queue, 30);
	enqueue(queue, 20);
	enqueue(queue, 40);

	cout << dequeue(queue) << " dequeued from queue" << endl;

	cout << "Front item is " << front(queue) << endl;
	cout << "Rear item is " << rear(queue) << endl;

	list_queue(queue);

	enqueue(queue, 40);
	enqueue(queue, 30);
	enqueue(queue, 20);
	enqueue(queue, 10);

	sortQueue(queue);
	list_queue(queue);

	return 0;
}
```

b. Stack with all its methods (push, pop, sort, etc.)

```
#include <iostream>
#include <stdio.h>
#include <stack>
#include <list>
using namespace std;

//Sıralı stack fonksiyonu
stack<int> sortStack(stack<int>& instack)
{
	stack<int> tmpStack;
	while (!instack.empty())
	{
		int tmp = instack.top();
		instack.pop();

		while (!tmpStack.empty() && tmpStack.top() < tmp)
		{
			instack.push(tmpStack.top());
			tmpStack.pop();
		}

		tmpStack.push(tmp);
	}

	return tmpStack;
}

//Tersten sıralı stack fonksiyonu
stack<int> sortStackReverse(stack<int>& input)
{
	stack<int> tmpStack;

	while (!input.empty())
	{

		int tmp = input.top();
		input.pop();

		while (!tmpStack.empty() && tmpStack.top() > tmp)
		{
			input.push(tmpStack.top());
			tmpStack.pop();
		}

		tmpStack.push(tmp);
	}

	return tmpStack;
}

int main()
{
	//stack oluşturma
	stack<int> input;
	////push işlemi
	input.push(30);
	input.push(6);
```

```
	input.push(21);
	input.push(75);
	input.push(14);
	input.push(7);

	////listeleme
	stack<int> tmpStack = sortStack(input);
	cout << "Sorted numbers are:" << endl;
	while (!tmpStack.empty())
	{
		cout << tmpStack.top() << " ";
		tmpStack.pop();
	}
	cout << endl;

	////push işlemi
	input.push(30);
	input.push(6);
	input.push(21);
	input.push(75);
	input.push(14);
	input.push(7);

	////reverse listeleme
	stack<int> tmpStack2 = sortStackReverse(input);
	cout << "Reverse sorted numbers are:" << endl;
	while (!tmpStack2.empty())
	{
		cout << tmpStack2.top() << " ";
		tmpStack2.pop();
	}
}
```